MATLAB-Python-Blender Pipeline Technical Report

Adam Aufderheide and Timothy Goulet

May 2025

This respository contains all files necessary to render images in a Blender model from a pose specified in MATLAB.

To run this script, download the MATLAB script and .blend file. Open the .blend file and navigate to the Python API script. Verify that the filepath for where images will be stored exists, then run the Blender Python API script. Then, go to MATLAB and run the script. You can view Blender render progress in the System Console, and messages will be sent from Blender back to the MATLAB command window for status updates. The rendered image will be in the specified file path, which will also be provided in a confirmation message posted in the MATLAB command window.

1 Introduction

In order to evaluate visual navigation algorithms, it is necessary to test them thoroughly using real or rendered images of the expected environment. This process often uses a large number of images, and consequently runs through the algorithm, which can be both time-consuming and computationally intensive. This rigorous testing process is essential in testing visual navigation algorithms due to the nature of the errors relating to the data association process, which are non-linear and non-gaussian, making them hard to predict and understand without testing the algorithm.

When real images of the environment are unavailable, rendered images are the best path. This is a common case for extraterrestrial terrain, where often no images or images of the desired quality of a certain terrain exist. Blender is powerful software for rendering images and creating models. It is open source, and unlike Unreal Engine, it can run without using a high-end GPU (essentially on any modern computer). Python is perfect for data association, as it has a variety of cutting-edge data-association programs. Blender also has a built-in Python API that allows it to be controlled by a Python script within the program. Therefore, it is optimal when using MATLAB to delegate data association to Python and image rendering to Blender. This is an outline of a pipeline between MATLAB, Python, and Blender to automate the image generation and data association process. MATLAB sends a pose vector to Blender, which renders an image, passes it to Python for data association, and returns the labeled feature coordinates to MATLAB. This pipeline extends the capabilities of preexisting MATLAB visual navigation programs.

2 Installing Software

How to install programs/git repository, program versions we used,

The version of Blender used in this project was 4.2.2. This can be found and downloaded at:

https://www.blender.org/download/releases/4-2/

The Python script is utilized using the API within Blender. Blender has its own embedded Python interpreter, so it is not necessary to have Python installed separately on your computer.

The version of MATLAB used in this project was R2023b, which can be found at: Mathworks.com

The scripts for the pipeline and the model used to run the sample run can be obtained by cloning the repository at:

https://gitlab.com/timgoulet/blender-sim-pipeline

Next we must install openCV directly in the Python API. To do this, open the terminal on your computer (not the one within the Python setup) and paste this command in:

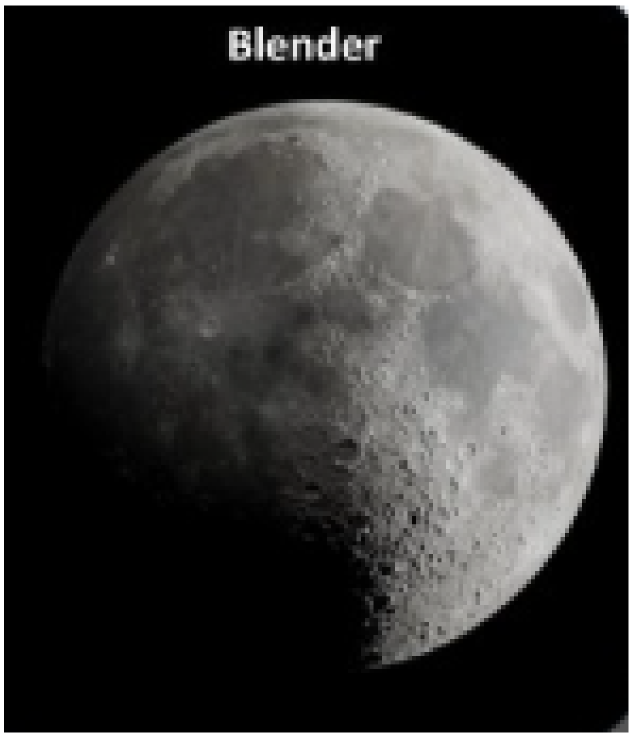
```
‘‘C:\Program Files\Blender Foundation\Blender 4.2\4.2\python\bin\python.exe’’
    -m pip install opencv-python
```

3 Pipeline Overview

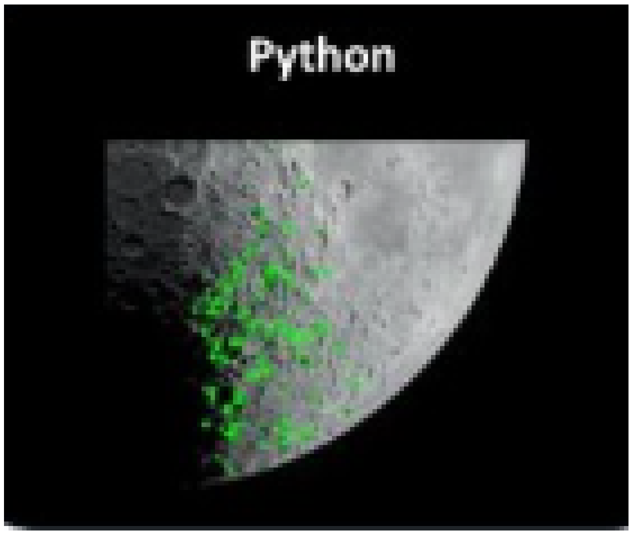
The repository mentioned in section 2 contains all files necessary to render images in a Blender model from a pose specified in MATLAB. To run the pipeline, download the MATLAB script and .blend file. the .blend file contains the Python script inside in the Python API as well as the sample model of the moon, created from LROC data and images. Open the .blend file and navigate to the Python API script. Verify that the filepath for where images will be stored exists. If it does not, or if you would like the images to go somewhere else, change the filepath within the script. Then run the Blender Python API script. This script should continue running and wait for a signal from MATLAB to continue. Do not turn this off before running the MATLAB program or it will not work. Next, open MATLAB and navigate to the .m file. Input the pose you would like Blender to render the image or images at, and then run the script. You can view Blender render progress in the System Console, and messages will be sent from Blender back to the MATLAB command window for status updates. The rendered image will be in the specified file path, which will also be provided in a confirmation message posted in the MATLAB command window once it has finished. The program may take around a minute to run.

4 Sample Run

This depicts a sample run of the pipeline. First, the vector is entered into MATLAB as input, and the program is run. The program in the Python/Blender connection is then started. This is an example of an image that could be returned through MATLAB:



Finally, the data association process is completed, and landmarks are marked out on the previous rendered image. Here is an example of what that image may look like:



5 Conclusions

To conclude, this report summarizes a pipeline to test visual navigation algorithms. This has a broad impact as there is a wide variety of use cases that this could pertain to. Visual navigation is not only used in the aerospace field (for which this was initially intended) but in robotics, autonomous vehicles, and many other fields as well. This pipeline streamlines the interface between programs and allows for easier testing. This will allow, in particular, the testing of vSLAM algorithms on extraterrestrial terrain, for the purpose of craft landing and navigation. This is a potential avenue for future work that could have a large impact on the aerospace industry and space exploration in general.